Recrutement des chargé·es de recherche CNRS 2024 – Concours 06/02
Liens entre différents pans de l'informatique théorique : automates, logique, théorie des langages de programmation

Lê Thành Dũng (Tito) Nguyễn – post-doctorant au LIP (ÉNS Lyon)

12 avril 2024

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\begin{aligned} \textit{langage rationnel} &\iff \text{défini par regex} \\ &\iff \text{reconnu par automate} \end{aligned}$$

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\begin{aligned} \textit{langage rationnel} &\iff \text{défini par regex} \\ &\iff \text{reconnu par automate} \end{aligned}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d’optimisation bas niveau
[Soyez-Martin ’23 ; Gienieczko, Murlak & Paperman ’24]

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\textit{langage rationnel} \iff \text{défini par regex} \iff \text{reconnu par automate}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves$\leftrightarrow$programmes)

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\begin{aligned} \textit{langage rationnel} &\iff \text{défini par regex} \\ &\iff \text{reconnu par automate} \end{aligned}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves$\leftrightarrow$programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\begin{aligned} \textit{langage rationnel} &\iff \text{défini par regex} \\ &\iff \text{reconnu par automate} \end{aligned}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves$\leftrightarrow$programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés
- contrôle de ressources (Rust)
  $\rightsquigarrow$ λ-calculs linéaires $\leftrightarrow$ **logique linéaire**

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\textit{langage rationnel} \iff \text{défini par regex} \iff \text{reconnu par automate}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves↔programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés
- contrôle de ressources (Rust)
  $\rightsquigarrow$ λ-calculs linéaires $\leftrightarrow$ **logique linéaire**

Complexité implicite : caractériser P, NP, PSPACE, ...
par des langages de prog. « de haut niveau »
$\rightsquigarrow$ analyse statique de complexité, optimisation, ...

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\textit{langage rationnel} \iff \text{défini par regex} \iff \text{reconnu par automate}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves$\leftrightarrow$programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés
- contrôle de ressources (Rust)
  $\rightsquigarrow$ λ-calculs linéaires $\leftrightarrow$ **logique linéaire**

Complexité implicite : caractériser P, NP, PSPACE, ...
par des langages de prog. « de haut niveau »
$\rightsquigarrow$ analyse statique de complexité, optimisation, ...

pour un langage $L \subseteq \Sigma^*$ (mots sur l'alphabet Σ), par exemple :

L programmable (« `string` $\to$ `bool` ») en *Light Linear Logic* $\iff$ $L \in \mathsf{P}$ [Girard 1998]

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\textit{langage rationnel} \iff \text{défini par regex} \iff \text{reconnu par automate}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves↔programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés
- contrôle de ressources (Rust)
  $\rightsquigarrow$ λ-calculs linéaires $\leftrightarrow$ **logique linéaire**

Complexité implicite : caractériser P, NP, PSPACE, ...
par des langages de prog. « de haut niveau »
$\rightsquigarrow$ analyse statique de complexité, optimisation, ...

pour un langage $L \subseteq \Sigma^*$ (mots sur l'alphabet Σ),

L programmable (« `string` $\to$ `bool` ») en λ-*calcul simplement typé* $\iff$

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\textit{langage rationnel} \iff \text{défini par regex} \iff \text{reconnu par automate}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves$\leftrightarrow$programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés
- contrôle de ressources (Rust)
  $\rightsquigarrow$ λ-calculs linéaires $\leftrightarrow$ **logique linéaire**

Complexité implicite : caractériser P, NP, PSPACE, ...
par des langages de prog. «de haut niveau»
$\rightsquigarrow$ analyse statique de complexité, optimisation, ...

Théorème [Hillebrand & Kanellakis 1996] : pour un langage $L \subseteq \Sigma^*$ (mots sur l'alphabet Σ),

L programmable («`string` $\to$ `bool`») en λ-*calcul simplement typé* $\iff$ L est un langage rationnel

Thèmes de recherche principaux

Thème 1 : Automates

Expressions régulières (regex) : motifs textuels, par ex.

$$b^*(ab^*ab^*)^* = \text{«nombre pair de “}a\text{”»}$$

traduction en *automates* $\rightsquigarrow$ filtrage efficace (`grep`)

$$\textit{langage rationnel} \iff \text{défini par regex} \iff \text{reconnu par automate}$$

sous-classes, par ex. *langages sans étoile* (années 1960)
$\rightsquigarrow$ possibilités d'optimisation bas niveau
[Soyez-Martin '23 ; Gienieczko, Murlak & Paperman '24]

Thème 2 : λ-calcul

théorie de la *programmation fonctionnelle*, liée à la *logique* (correspondance preuves↔programmes)

- typage statique (OCaml, Haskell, ...)
  $\rightsquigarrow$ λ-calculs typés
- contrôle de ressources (Rust)
  $\rightsquigarrow$ λ-calculs linéaires $\leftrightarrow$ **logique linéaire**

Complexité implicite : caractériser P, NP, PSPACE, ...
par des langages de prog. « de haut niveau »
$\rightsquigarrow$ analyse statique de complexité, optimisation, ...

Théorème [Hillebrand & Kanellakis 1996] : pour un langage $L \subseteq \Sigma^*$ (mots sur l'alphabet Σ),

L programmable (« `string` $\to$ `bool` ») en λ-*calcul simplement typé* $\iff$ L est un langage rationnel

$\longrightarrow$ théorie des *automates implicites* : lancée dans ma thèse

2012–18 Élève ÉNS Paris

2012–18 Élève ÉNS Paris

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle $\longrightarrow$ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » ($\sim$ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal $\times 2$

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle $\longrightarrow$ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » ($\sim$ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal $\times 2$

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle $\longrightarrow$ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » ($\sim$ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal $\times 2$

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » ($\sim$ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal $\times 2$

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Liens automates $\leftrightarrow$ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse (avec Pradic, $>$ 100 pages) •

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » ($\sim$ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal $\times 2$

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul

Liens automates $\leftrightarrow$ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse (avec Pradic, $>$ 100 pages) •

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

09/21–02/22 Ingénieur CNRS/IRISA

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » ($\sim$ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal $\times 2$

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  $\rightsquigarrow$ motivé par le λ-calcul

Liens automates $\leftrightarrow$ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse (avec Pradic, $>$ 100 pages) •

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

09/21–02/22 Ingénieur CNRS/IRISA

03/22–08/22 Postdoc LIX

09/22–... Postdoc LIP

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » (∼ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal ×2

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul

Liens automates ↔ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse (avec Pradic, > 100 pages) ●

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

09/21–02/22 Ingénieur CNRS/IRISA

03/22–08/22 Postdoc LIX

09/22–... Postdoc LIP

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » (∼ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal ×2

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul

Liens automates ↔ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse (avec Pradic, > 100 pages) •

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19
- N., accepté à LMCS
  ⇝ utilise des automates
- Das, Mazza, N. & Zeilberger, annoncé à TLLA'23

(au LIX : ANR LambdaComb)

2012–18 Élève ÉNS Paris
2018–21 (Pré)-doctorat LIPN
09/21–02/22 Ingénieur CNRS/IRISA
03/22–08/22 Postdoc LIX
09/22–... Postdoc LIP

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » (∼ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal ×2

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul
- Bojańczyk & N., ICALP'23

+ 3 soumissions en cours
(FI, DMTCS, ICALP'24)

Liens automates ↔ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse
(avec Pradic, > 100 pages) •

- Moreau & N., CSL'24
- N. & Vanoni, soumis

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19
- N., accepté à LMCS
  ⇝ utilise des automates
- Das, Mazza, N. & Zeilberger,
  annoncé à TLLA'23

(au LIX : ANR LambdaComb)

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

09/21–02/22 Ingénieur CNRS/IRISA

03/22–08/22 Postdoc LIX

09/22–... Postdoc LIP

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » (∼ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal ×2

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul
- Bojańczyk & N., ICALP'23

+ 3 soumissions en cours
(FI, DMTCS, ICALP'24)

• = *théorie des catégories*

Liens automates ↔ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse
(avec Pradic, > 100 pages) •

- Moreau & N., CSL'24
- N. & Vanoni, soumis

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19
- N., accepté à LMCS
  ⇝ utilise des automates
- Das, Mazza, N. & Zeilberger,
  annoncé à TLLA'23

(au LIX : ANR LambdaComb)

2012–18 Élève ÉNS Paris

2018–21 (Pré)-doctorat LIPN

09/21–02/22 Ingénieur CNRS/IRISA

03/22–08/22 Postdoc LIX

09/22–... Postdoc LIP

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » (∼ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal ×2

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul
- Bojańczyk & N., ICALP'23

+ 3 soumissions en cours
(FI, DMTCS, ICALP'24)

• = *théorie des catégories*

Liens automates ↔ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse
(avec Pradic, > 100 pages) •

- Moreau & N., CSL'24
- N. & Vanoni, soumis

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19
- N., accepté à LMCS
  ⇝ utilise des automates
- Das, Mazza, N. & Zeilberger,
  annoncé à TLLA'23

(au LIX : ANR LambdaComb)

Comité de programme *Trends in Linear Logic & Applications* (TLLA) 2022

2012–18 Élève ÉNS Paris
2018–21 (Pré)-doctorat LIPN
09/21–02/22 Ingénieur CNRS/IRISA
03/22–08/22 Postdoc LIX
09/22–... Postdoc LIP

Application de l'algorithmique des graphes à la logique linéaire

- M1 logique, M2 recherche opérationnelle ⟶ projet perso
- mènera à réfuter la conjecture « pomset logic = BV » (∼ 20 ans)

[N. FSCD'18 ; N. & Straßburger, CSL'22] + version journal ×2

Pure logique linéaire : [N. & Seiller (dir. thèse), post-proceedings TLLA'18]

Pure théorie des automates

- N., Noûs & Pradic, ICALP'21
  ⇝ motivé par le λ-calcul
- Bojańczyk & N., ICALP'23

+ 3 soumissions en cours
(FI, DMTCS, ICALP'24)

• = *théorie des catégories*

Liens automates ↔ λ-calcul

- N., post-proc. DICE-FOPARA'19
- N. & Pradic, ICALP'20

+ résultats non publiés de ma thèse
(avec Pradic, > 100 pages) •

- Moreau & N., CSL'24
- N. & Vanoni, soumis

Complexité en λ-calcul (linéaire)

- N. & Pradic ICALP'19
- N., accepté à LMCS
  ⇝ utilise des automates
- Das, Mazza, N. & Zeilberger,
  annoncé à TLLA'23

(au LIX : ANR LambdaComb)

Comité de programme *Trends in Linear Logic & Applications* (TLLA) 2022

Collaborations+visites UK (**Cécilia Pradic**, Anupam Das, Sandra Kiefer) & Pologne (M. Bojańczyk, R. Stefański)

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini ($\approx$ automate déterministe)

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...

Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...

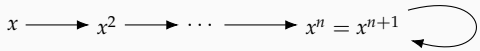
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)



Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

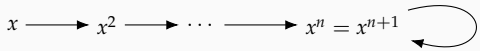
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)



Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

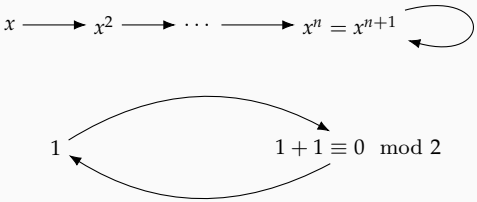
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)



Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

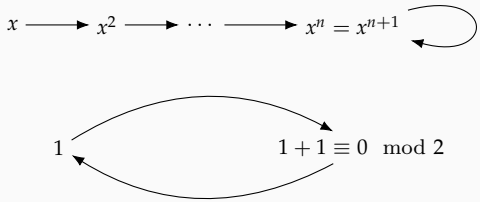
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini ($\approx$ automate déterministe)



Imposer l'apériodicité dans un langage de programmation ?

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

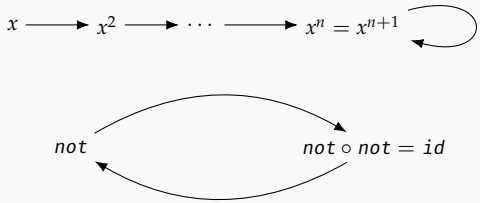
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini ($\approx$ automate déterministe)



Imposer l'apériodicité dans un langage de programmation ?
$\texttt{not}^{n+1} \neq \texttt{not}^n$

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

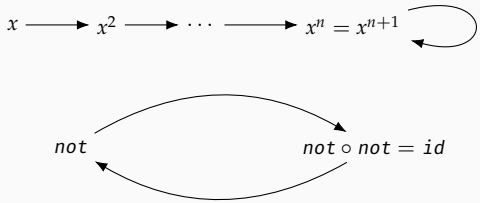
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)



Imposer l'apériodicité dans un langage de programmation ?
$\texttt{not}^{n+1} \neq \texttt{not}^n \rightsquigarrow$ on doit exclure $\texttt{not} : \texttt{Bool} \to \texttt{Bool}$

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

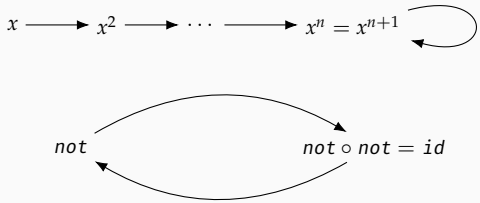
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)



Imposer l'apériodicité dans un langage de programmation ?
$\texttt{not}^{n+1} \neq \texttt{not}^n \rightsquigarrow$ on doit exclure `not` : `Bool` $\to$ `Bool`

En λ-calcul : codage fonctionnel des booléens

$b(x, y) \approx$ « `if b then x else y` »

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

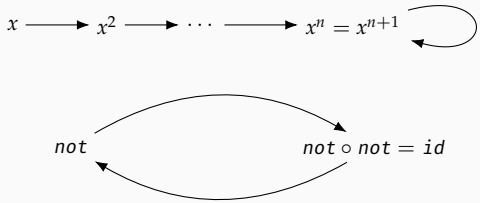
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ « *reconnu* » *par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini
($\approx$ automate déterministe)



Imposer l'apériodicité dans un langage de programmation ?
$\texttt{not}^{n+1} \neq \texttt{not}^n \rightsquigarrow$ on doit exclure $\texttt{not} : \texttt{Bool} \to \texttt{Bool}$

En λ-calcul : codage fonctionnel des booléens

$b(x, y) \approx$ « `if b then x else y` »

d'où $\texttt{not}(b) = [(x, y) \mapsto b(y, x)]$

Un résultat de ma thèse : langages sans étoile en λ-calcul (1/2)

Définition

Langage sans étoile : défini par expression régulière

- sans étoile de répétition L^*
- avec *complémentation* : $L^c = \Sigma^* \setminus L$

$(ab)^* = (b\varnothing^c|\varnothing^c a|\varnothing^c aa\varnothing^c|\varnothing^c bb\varnothing^c)^c \quad (\Sigma = \{a, b\})$

Mais $(aa)^*$ n'a pas d'expression sans étoile...
son « monoïde minimal » est $\mathbb{Z}/2\mathbb{Z}$: *pas apériodique*

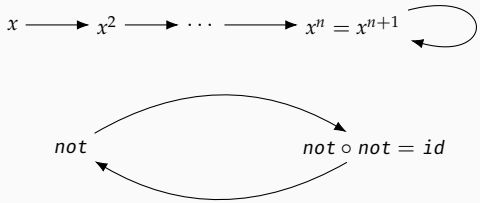
Définition

Monoïde M : on peut « composer » les éléments
Apériodique : $\forall x \in M,\ \exists n \in \mathbb{N} : x^n = x^{n+1}$

Théorème (Schützenberger 1965)

Langage sans étoile $\iff$ *« reconnu » par un monoïde fini* apériodique

Classique : rationnel $\Leftrightarrow$ reconnu par monoïde fini ($\approx$ automate déterministe)



Imposer l'apériodicité dans un langage de programmation ?
$\texttt{not}^{n+1} \neq \texttt{not}^n \rightsquigarrow$ on doit exclure $\texttt{not} : \texttt{Bool} \to \texttt{Bool}$

En λ-calcul : codage fonctionnel des booléens

$b(x, y) \approx$ « `if b then x else y` »

d'où $\texttt{not}(b) = [(x, y) \mapsto b(y, x)]$

Idée : l'ordre des arguments importe

« une fonction $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y, puisque x est pris en argument avant y »
$\longrightarrow$ *λ-calcul non commutatif*

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Faisons coexister les fonctions affines $A \multimap B$
et non restreintes $A \Rightarrow B$

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Faisons coexister les fonctions affines $A \multimap B$
et non restreintes $A \Rightarrow B$

Utilisation d'un système $\simeq$ [Polakow & Pfenning 1999]

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Faisons coexister les fonctions affines $A \multimap B$
et non restreintes $A \Rightarrow B$

Utilisation d'un système $\simeq$ [Polakow & Pfenning 1999]

Le théorème principal [N. & Pradic, ICALP'20]

Le langage $L \subseteq \Sigma^*$ est *sans étoile*
$\iff$
L est défini par un programme $\Sigma^* \multimap \texttt{Bool}$
dans ce λ-calcul affine non commutatif
(modulo condition d'affinité « cachée » dans l'entrée)

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Faisons coexister les fonctions affines $A \multimap B$
et non restreintes $A \Rightarrow B$

Utilisation d'un système $\simeq$ [Polakow & Pfenning 1999]

Le théorème principal [N. & Pradic, ICALP'20]

Le langage $L \subseteq \Sigma^*$ est *sans étoile*
$\Longleftrightarrow$
L est défini par un programme $\Sigma^* \multimap \texttt{Bool}$
dans ce λ-calcul affine non commutatif
(modulo condition d'affinité « cachée » dans l'entrée)

Cas commutatif : langages rationnels
$\longrightarrow$ véritable effet calculatoire de la non-comm.

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Faisons coexister les fonctions affines $A \multimap B$
et non restreintes $A \Rightarrow B$

Utilisation d'un système $\simeq$ [Polakow & Pfenning 1999]

Le théorème principal [N. & Pradic, ICALP'20]

Le langage $L \subseteq \Sigma^*$ est *sans étoile*
$\iff$
L est défini par un programme $\Sigma^* \multimap \texttt{Bool}$
dans ce λ-calcul affine non commutatif
(modulo condition d'affinité « cachée » dans l'entrée)

Cas commutatif : langages rationnels
$\longrightarrow$ véritable effet calculatoire de la non-comm.

Ma thèse : caractérisations de fonctions entre mots en λ-calcul affine commutatif

- $\Sigma^* \multimap \Gamma^*$: classe connue des transductions MSO (cas non commutatif : transductions FO)

par ex. $w_1 \# \ldots \# w_n \mapsto w_n w_n \# \ldots \# w_1 w_1$

Un résultat de ma thèse : langages sans étoile en λ-calcul (2/2)

Idée 1 : l'ordre des arguments importe

« $(x, y) \mapsto \ldots$ ne doit pas utiliser x après y »

Pour des raisons techniques, la non-commutativité ne peut concerner que des arguments *affines* au sens de la **logique linéaire** [Girard 1987]

Idée 2 : interdire la duplication

« une fonction utilise son argument *au plus une fois* »
$\longrightarrow$ λ-calcul *affine* non commutatif

Faisons coexister les fonctions affines $A \multimap B$
et non restreintes $A \Rightarrow B$

Utilisation d'un système $\simeq$ [Polakow & Pfenning 1999]

Le théorème principal [N. & Pradic, ICALP'20]

Le langage $L \subseteq \Sigma^*$ est *sans étoile*
$\iff$
L est défini par un programme $\Sigma^* \multimap \texttt{Bool}$
dans ce λ-calcul affine non commutatif
(modulo condition d'affinité « cachée » dans l'entrée)

Cas commutatif : langages rationnels
$\longrightarrow$ véritable effet calculatoire de la non-comm.

Ma thèse : caractérisations de fonctions entre mots en λ-calcul affine commutatif

- $\Sigma^* \multimap \Gamma^*$: classe connue des transductions MSO (cas non commutatif : transductions FO)

  par ex. $\quad w_1 \# \ldots \# w_n \mapsto w_n w_n \# \ldots \# w_1 w_1$

- $\Sigma^* \Rightarrow \Gamma^*$: *nouvelle* classe des fonctions « polyrégulières sans comparaison », par ex. $w \mapsto \underbrace{w \ldots w}_{|w| \text{ fois}}$

  $\rightsquigarrow$ [N., Noûs & Pradic, ICALP'21] : pure théorie des automates

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

$$\underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine } + \text{ [N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}}$$

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct° rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine } + \text{ [N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct}^\circ \text{ rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en λ-*calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct° rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en λ-*calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct° rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en λ-*calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct° rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en λ-*calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]
- $L \subseteq \Sigma^*$ (càd $\chi_L : \Sigma^* \to$ Bool) à états finis = lang. rat.

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct}^\circ \text{ rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en λ-*calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]
- $L \subseteq \Sigma^*$ (càd $\chi_L : \Sigma^* \to$ Bool) à états finis = lang. rat.
- **Fonctions $\Sigma^* \to \Gamma^*$ à états finis : ???**

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct}^\circ \text{ rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en λ-*calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]
- $L \subseteq \Sigma^*$ (càd $\chi_L : \Sigma^* \to$ Bool) à états finis = lang. rat.
- **Fonctions $\Sigma^* \to \Gamma^*$ à états finis : ???**

Hypothèse « morale » [Bojańczyk]

États finis $\cap$ efficace = polyrégulières

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct}^\circ \text{ rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en *λ-calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]
- $L \subseteq \Sigma^*$ (càd $\chi_L : \Sigma^* \to$ Bool) à états finis = lang. rat.
- **Fonctions $\Sigma^* \to \Gamma^*$ à états finis : ???**

Hypothèse « morale » [Bojańczyk]

États finis $\cap$ efficace = polyrégulières

Comme $\Lambda = +$ grande classe « à états finis » connue :

Conjecture

$\Lambda \cap$ temps polynomial = polyrégulières

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct}^\circ \text{ rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en *λ-calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]
- $L \subseteq \Sigma^*$ (càd $\chi_L : \Sigma^* \to$ Bool) à états finis = lang. rat.
- **Fonctions $\Sigma^* \to \Gamma^*$ à états finis : ???**

Hypothèse « morale » [Bojańczyk]

États finis $\cap$ efficace = polyrégulières

Comme Λ = + grande classe « à états finis » connue :

Conjecture

$\Lambda \cap$ temps polynomial = polyrégulières

Appartenance à 1 sous-classe vs *vitesse de croissance* : cf. [Kiefer, N. & Pradic 2023 ; Gallot, Lhote & N., en prép.]

Projet de recherche : calculabilité par états finis, optimisation et λ-calcul

Fonctions $f\colon \Sigma^* \to \Gamma^*$, calculées par des *transducteurs* (automates à sortie), **t.q.** $L \subseteq \Sigma^*$ **rationnel** $\Rightarrow f^{-1}(L)$ **rat.** :

$$\text{fct}^\circ \text{ rationnelles} \subsetneq \underbrace{\text{transductions MSO}}_{\Sigma^* \multimap \Gamma^* \text{ affine}} \subsetneq \underbrace{\text{polyrégulières sans comparaisons}}_{\Sigma^* \Rightarrow \Gamma^* \text{ affine} + \text{[N., Noûs \& Pradic ICALP'21]}} \subsetneq \underbrace{\text{polyrégulières}}_{\text{[Bojańczyk 2018]}} \subsetneq \dots$$

$\Lambda = (\Sigma^* \Rightarrow \Gamma^*$ en *λ-calcul simplement typé*$)$

fonctions polyrégulières $\subsetneq \Lambda$

(polynomial) (hyper-exponentiel)

Ligne directrice : une question ouverte, imprécise

Notion *absolue* de *calculabilité par états finis* ? (c'est-à-dire par des modèles « raisonnables » de transducteurs)

- Calculabilité tout court : « thèse de Church–Turing »
- *Efficace* : temps polynomial [Cobham, Edmonds 1960s]
- $L \subseteq \Sigma^*$ (càd $\chi_L : \Sigma^* \to$ Bool) à états finis = lang. rat.
- **Fonctions $\Sigma^* \to \Gamma^*$ à états finis : ???**

Hypothèse « morale » [Bojańczyk]

États finis $\cap$ efficace = polyrégulières

Comme Λ = + grande classe « à états finis » connue :

Conjecture

$\Lambda \cap$ temps polynomial = polyrégulières

Appartenance à 1 sous-classe vs *vitesse de croissance* :
cf. [Kiefer, N. & Pradic 2023 ; Gallot, Lhote & N., en prép.]
...mais les preuves existantes ne s'adaptent pas à Λ

$\to$ défi pour la communauté logique linéaire

sémantiques quantitatives à la mode $\rightsquigarrow$ outils adaptés ?

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \text{Bool}$ langages rationnels

$f\colon \Sigma^* \to \Gamma^*$ **quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de λ-*calculs typés*

À long terme : « États finis $\cap$ efficace = polyrégulières » ?

$\rightsquigarrow$ nécessite de nouveaux outils

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \mathrm{Bool}$	langages rationnels
$f\colon \Sigma^* \to \Gamma^*$	**quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de λ-*calculs typés*

À long terme : « États finis $\cap$ efficace = polyrégulières » ?
$\rightsquigarrow$ nécessite de nouveaux outils

À court terme : liens étroits avec le *higher-order model checking*
(vérification de programmes fonctionnels)

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \mathrm{Bool}$ langages rationnels

$f\colon \Sigma^* \to \Gamma^*$ **quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de λ-*calculs typés*

À long terme : « États finis $\cap$ efficace = polyrégulières » ?
$\rightsquigarrow$ nécessite de nouveaux outils

À court terme : liens étroits avec le *higher-order model checking*
(vérification de programmes fonctionnels)

Λ vs fonctions provenant par ex. de la théorie des modèles
(« interprétations MSO ensemblistes »)

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \text{Bool}$ langages rationnels

$f\colon \Sigma^* \to \Gamma^*$ **quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de *λ-calculs typés*

À long terme : « États finis $\cap$ efficace = polyrégulières » ?
$\rightsquigarrow$ nécessite de nouveaux outils

À court terme : liens étroits avec le *higher-order model checking*
(vérification de programmes fonctionnels)

Λ vs fonctions provenant par ex. de la théorie des modèles
(« interprétations MSO ensemblistes »)

+ poursuivre les « automates implicites »
avec C. Pradic, par ex. sur les mots infinis
(en cours avec A. De, C. Grellois, D. Kuperberg)

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \text{Bool}$ langages rationnels

$f\colon \Sigma^* \to \Gamma^*$ **quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de λ-*calculs typés*

À long terme : « États finis $\cap$ efficace = polyrégulières » ?
$\rightsquigarrow$ nécessite de nouveaux outils

À court terme : liens étroits avec le *higher-order model checking*
(vérification de programmes fonctionnels)

Λ vs fonctions provenant par ex. de la théorie des modèles
(« interprétations MSO ensemblistes »)

+ poursuivre les « automates implicites »
avec C. Pradic, par ex. sur les mots infinis
(en cours avec A. De, C. Grellois, D. Kuperberg)

Continuer les découvertes imprévues par ex.

- « pomset logic $\neq$ BV » via algo graphes
- complexité en λ-calcul safe via automates

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \text{Bool}$ langages rationnels
$f\colon \Sigma^* \to \Gamma^*$ **quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de λ-*calculs typés*

À long terme : « États finis ∩ efficace = polyrégulières » ?
$\rightsquigarrow$ nécessite de nouveaux outils

À court terme : liens étroits avec le *higher-order model checking*
(vérification de programmes fonctionnels)
Λ vs fonctions provenant par ex. de la théorie des modèles
(« interprétations MSO ensemblistes »)

+ poursuivre les « automates implicites »
avec C. Pradic, par ex. sur les mots infinis
(en cours avec A. De, C. Grellois, D. Kuperberg)

Continuer les découvertes imprévues par ex.

- « pomset logic $\neq$ BV » via algo graphes
- complexité en λ-calcul safe via automates

Thématiques sociales et environnementales

- aide à l'organisation de la 1ère édition d'**Undone Computer Science** (fév. 2024)
- à terme : diversifier ma recherche ?

Projet de recherche : calculabilité par états finis

$\Sigma^* \to \mathrm{Bool}$ langages rationnels
$f\colon \Sigma^* \to \Gamma^*$ **quelle classe maximale raisonnable ?**

Parti pris : étudier f provenant de λ-*calculs typés*

À long terme : « États finis ∩ efficace = polyrégulières » ?
$\rightsquigarrow$ nécessite de nouveaux outils

À court terme : liens étroits avec le *higher-order model checking*
(vérification de programmes fonctionnels)
Λ vs fonctions provenant par ex. de la théorie des modèles
(« interprétations MSO ensemblistes »)

+ poursuivre les « automates implicites »
avec C. Pradic, par ex. sur les mots infinis
(en cours avec A. De, C. Grellois, D. Kuperberg)

Continuer les découvertes imprévues par ex.

- « pomset logic $\neq$ BV » via algo graphes
- complexité en λ-calcul safe via automates

Thématiques sociales et environnementales

- aide à l'organisation de la 1ère édition d'**Undone Computer Science** (fév. 2024)
- à terme : diversifier ma recherche ?

Intégration :

- LIP : déjà postdoc $\to$ dialogue à la fois avec les côtés automates et logique linéaire dans l'équipe Plume
- LIS : double affectation Move $\to$ automates – collaboration en cours avec N. Lhote
  LSC $\to$ λ-calcul – convergences possibles avec P. Clairambault & R. Crubillé
  + liens avec L. Santocanale via ANR LambdaComb
- IRIF : interface automates/λ-calcul $\to$ S. van Gool & P.-A. Melliès (collaboration avec leur doctorant V. Moreau)
  automates/catégories $\to$ T. Colcombet & D. Petrişan (travaux utilisés dans ma thèse)
  + transducteurs (O. Carton, S. Winter) + logique linéaire (beaucoup de monde)